Assumption-Based Runtime Verification with Partial Observability and Resets

(RV 2019 regular paper)

Alessandro Cimatti [1] Chun Tian [1,2] Stefano Tonetta [1]

{cimatti,ctian,tonettas}@fbk.eu

[1]Fondazione Bruno Kessler, Italy

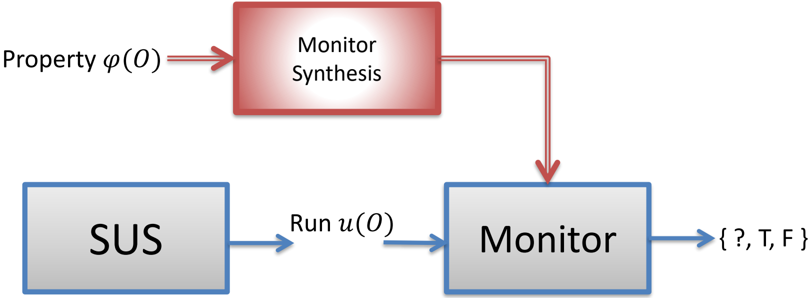
[2]University of Trento, Italy

October 2019

Outline

Runtime Verification (RV)

- A lightweight verification technique providing checking if a system under scrutiny (SUS) satisfies/violates a monitoring property.
- Focus on a single *finite* trace instead of all traces from SUS.
- Has an *incremental* fashion, outputting verdicts on each input state.
- Applicable to *black box* systems where a model is not available.
- Assumes full observability usually.

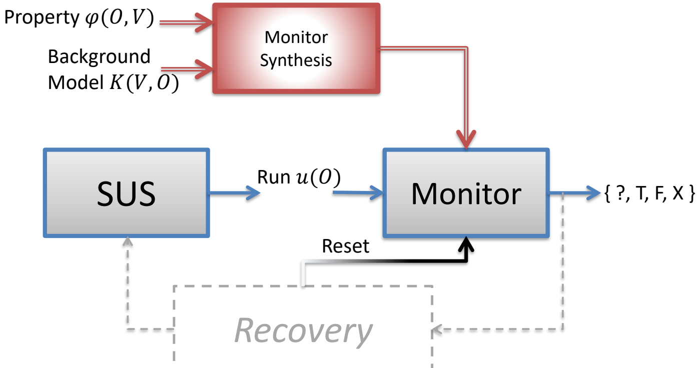


Assumption-Based Runtime Verification

However, one *almost always* knows something about the SUS, e.g.

- Models produced during the system design;
- Interaction with system operators (i.e. people) — domain knowledge.
- Mathematics/physical principles, e.g. $\varphi = (i \leqslant 5)\,\mathsf{U}\,(i > 10)$.

These knowledge may be leveraged to get *better* monitors.



Resettable Monitors

- Traditionally the monitor only evaluates $[\![u \models \varphi]\!]$ $(= [\![u, 0 \models \varphi]\!])$;
- The resettable monitor takes as input some *reset* signals that change the reference time of evaluating monitor properties, e.g.
  from $[\![u, i \models \varphi]\!]$ to $[\![u, j \models \varphi]\!]$ $(j > i)$;
- The execution history of SUS is preserved during resets, possible impacts to monitoring outputs:
  1. Under assumptions, the belief states after resets may be different with initial belief states;
  2. With past operators, historical inputs may change the initial evaluation of a monitoring property.

The Motivation

1. Monotonic monitors: still meaningful after reaching conclusive verdicts (then being reset).
2. Monitoring Past-Time LTL (to be explained).

Outline

(Propositional) Linear Temporal Logic

Syntax ($p \in AP$)

$$\varphi ::= \mathrm{true} \mid p \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathsf{X}\,\varphi \mid \varphi\,\mathsf{U}\,\varphi \mid \mathsf{Y}\varphi \mid \varphi\,\mathsf{S}\,\varphi$$

- X stands for *next*, U for *until*, Y for *previous*, S for *since*.
- logical constants and operators like false, $\wedge$, $\rightarrow$ and $\leftrightarrow$ are used as syntactic sugars with the standard meaning.
- Abbreviations:

$$\begin{aligned}
\mathsf{F}\,\varphi &\doteq \mathrm{true}\,\mathsf{U}\,\varphi(\text{eventually}),\\
\mathsf{G}\,\varphi &\doteq \neg\mathsf{F}\,\neg\varphi(\text{globally}),\\
\mathsf{O}\,\varphi &\doteq \mathrm{true}\;\mathsf{S}\;\varphi(\text{once}),\\
\mathsf{H}\,\varphi &\doteq \neg\mathsf{O}\,\neg\varphi(\text{historically}).
\end{aligned}$$

Recall: LTL_3 semantics

- Three-valued semantics of LTL formula φ over a finite word $u \in \Sigma^*$:

$$[\![u, i \models \varphi]\!]_3 = \begin{cases} \top, & \text{if } \forall w \in \Sigma^\omega.\ u \cdot w, i \models \varphi, \\ \bot, & \text{if } \forall w \in \Sigma^\omega.\ u \cdot w, i \not\models \varphi, \\ ?, & \text{otherwise} \quad . \end{cases}$$

  with $[\![u \models \varphi]\!]_3$ denoting $[\![u, 0 \models \varphi]\!]_3$.
- $[\![u \models \varphi]\!]_3 = \top/\bot$ if all extensions of u satisfy/violate φ;
- Monitor construction:[1]

$$\mathcal{M}_\varphi(u) = [\![u \models \varphi]\!]_3 \quad .$$

[1] A. Bauer, M. Leucker, and C. Schallhart. *Runtime Verification for LTL and TLTL.* (2011)

ABRV-LTL semantics

Let $K \doteq \langle V_K, \Theta_K, \rho_K, \mathcal{J}_K \rangle$ be an FKS, φ be an LTL formula built from AP. Let $\psi(O) \in \Psi(O)^*$ be a finite sequence of Boolean formulae over $O \subseteq V_K \cup AP$. We also define

$$\mathcal{L}^K(\psi(O)) \doteq \left\{ w \in \mathcal{L}(K) \;\middle|\; \forall i.\; i < |\psi(O)| \Rightarrow w_i(V_K \cup AP) \models \psi_i(O) \right\}$$

to be the set of runs in K which are compatible with $\psi(O)$.

Definition

The ABRV-LTL semantics of φ over $\psi(O)$ under K is defined as

$$[\![\psi(O), i \models \varphi]\!]_4^K \doteq \begin{cases} \times, & \text{if } \mathcal{L}^K(\psi(O)) = \emptyset \\ \top^{\mathrm{a}}, & \text{if } \mathcal{L}^K(\psi(O)) \neq \emptyset \wedge \forall w \in \mathcal{L}^K(\psi(O)).\; w, i \models \varphi \\ \bot^{\mathrm{a}}, & \text{if } \mathcal{L}^K(\psi(O)) \neq \emptyset \wedge \forall w \in \mathcal{L}^K(\psi(O)).\; w, i \models \neg\varphi \\ ?, & \text{otherwise}\;. \end{cases}$$

ABRV-LTL verdicts (and the lattice)

$\mathbb{B}_4 \doteq \{\top^{\mathrm{a}}, \bot^{\mathrm{a}}, ?, \times\}$:

- *conclusive true* ($\top^{\mathrm{a}}$) (or *true under assumption*)
- *conclusive false* ($\bot^{\mathrm{a}}$) (or *false under assumption*)
- *inconclusive* ($?$)
- *out-of-model* ($\times$)

The lattice

$$\begin{tikzcd}
\top \\
? \arrow[u] \arrow[d] \\
\bot
\end{tikzcd}
\qquad\qquad
\begin{tikzcd}
 & \times & \\
\top^{\mathrm{a}} \arrow[ur] & & \bot^{\mathrm{a}} \arrow[ul] \\
 & ? \arrow[ul] \arrow[ur] &
\end{tikzcd}$$

LTL_3 ABRV-LTL

The ABRV Framework

- The input is enriched with resets: $u \in (\Psi(O) \times \mathbb{B})^*$.
- Each input state is a pair of an observation and a Boolean representing the reset;
- Given a property ϕ and an assumption K, the problem of *Assumption-based Runtime Verification (ABRV)* is to construct a function $\mathcal{M}_\varphi^K : (\Psi(O) \times \mathbb{B})^* \to \mathbb{B}_4$ such that

$$\mathcal{M}_\varphi^K(u) = [\![\textsc{obs}(u), \textsc{mrr}(u) \models \varphi]\!]_4^K$$

where

- $\textsc{obs}(\cdot)$ (observations) is the projection of u from $\Psi(O) \times \mathbb{B}$ to $\Psi(O)$,
- $\textsc{res}(\cdot)$ (resets) is the projection of u (or u_i) from $\Psi(O) \times \mathbb{B}$ to $\mathbb{B}$,
- $\textsc{mrr}(u)$ (the most recent reset) is the maximal i such that $\textsc{res}(u_i) = \top$.

Special Case: Runtime Verification of Past-Time LTL

Past-Time LTL (PtLTL)[2]: LTL with only past operators (Y, S).
Let $u = s_1\, s_2 \cdot s_n$ and $u_i = s_1\, s_2 \cdot s_i$:

$$
\begin{aligned}
u \models_p p &\Leftrightarrow p \in s_{n-1} \\
u \models_p \mathsf{Y}\,\varphi &\Leftrightarrow u_{n-1} \models_p \varphi \text{ (if } n > 1\text{) or } u \models_p \varphi \text{ (if } n = 1\text{)} \\
u \models_p \varphi\,\mathsf{S}\,\psi &\Leftrightarrow u_j \models_p \psi \; (1 \leqslant j \leqslant n) \text{ and } u_i \models_p \varphi \; (j < i \leqslant n)
\end{aligned}
$$

Convert PtLTL to ABRV-LTL (K is empty)

$$
\begin{aligned}
[\![u \models_p \varphi]\!] = \top &\quad\leftrightarrow\quad [\![u, |u| - 1 \models \varphi]\!]_4^K = \top^{\mathrm{a}}, \\
[\![u \models_p \varphi]\!] = \bot &\quad\leftrightarrow\quad [\![u, |u| - 1 \models \varphi]\!]_4^K = \bot^{\mathrm{a}} \;.
\end{aligned}
$$

$\mathrm{MRR}(u) = |u|$ or $\forall i.\, \mathrm{RES}(u_i) = \top$.

[2] K. Havelund and G. Roşu. Synthesizing Monitors for Safety Properties. In J.-P. Katoen and P. Stevens, editors, *LNCS 2280 - Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2002)*, pages 342–356. Springer, Berlin, Heidelberg, June 2013

Outline

Translating LTL to ω-automata (1)

Elementary Variables

$$
\begin{aligned}
\mathrm{el}(\mathrm{true}) &= \emptyset, & \mathrm{el}(\mathsf{X}\phi) &= \{\mathrm{X}_{\phi}\} \cup \mathrm{el}(\phi), \\
\mathrm{el}(p) &= \{p\}, & \mathrm{el}(\phi \mathsf{U} \psi) &= \{\mathrm{X}_{\phi \mathsf{U} \psi}\} \cup \mathrm{el}(\phi) \cup \mathrm{el}(\psi), \\
\mathrm{el}(\neg\phi) &= \mathrm{el}(\phi), & \mathrm{el}(\mathsf{Y}\phi) &= \{\mathrm{Y}_{\phi}\} \cup \mathrm{el}(\phi), \\
\mathrm{el}(\phi \vee \psi) &= \mathrm{el}(\phi) \cup \mathrm{el}(\psi), & \mathrm{el}(\phi \mathsf{S} \psi) &= \{\mathrm{Y}_{\phi \mathsf{S} \psi}\} \cup \mathrm{el}(\phi) \cup \mathrm{el}(\psi) \ .
\end{aligned}
$$

Expansion Laws (for the xNF conversion)

$$
\psi \,\mathsf{U}\, \phi \Leftrightarrow \phi \vee (\psi \wedge \mathsf{X}(\psi \,\mathsf{U}\, \phi)), \qquad \psi \,\mathsf{S}\, \phi \Leftrightarrow \phi \vee (\psi \wedge \mathsf{Y}(\psi \,\mathsf{S}\, \phi)) \ .
$$

Example (Translating LTL to Propositional Logic $\chi(\cdot)$)

$$
\chi(p \,\mathsf{U}\, q) = q \vee (p \wedge \mathrm{X}_{p \mathsf{U} q}) \ .
$$

Translating LTL to ω-automata (2)

NUXMV's tableau construction: $T_\varphi \doteq \langle V_\varphi, \Theta_\varphi, \rho_\varphi, \mathcal{J}_\varphi \rangle$, where

- Set of Boolean variables: $V_\varphi \doteq \mathrm{el}(\varphi)$,
- Initial condition:

$$\Theta_\varphi \doteq \chi(\varphi) \wedge \bigwedge_{\mathrm{Y}_\psi \in \mathrm{el}(\varphi)} \neg \mathrm{Y}_\psi ,$$

- Transition relation:

$$\rho_\varphi \doteq \bigwedge_{\mathrm{X}_\psi \in \mathrm{el}(\varphi)} \left(\mathrm{X}_\psi \leftrightarrow \chi'(\psi)\right) \ \wedge \bigwedge_{\mathrm{Y}_\psi \in \mathrm{el}(\varphi)} \left(\chi(\psi) \leftrightarrow \mathrm{Y}'_\psi\right),$$

- Justice set (a fairness condition):

$$\mathcal{J}_\varphi \doteq \left\{\chi(\psi \, \mathsf{U} \, \phi) \rightarrow \chi(\phi) \mid \mathrm{X}_{\psi \mathsf{U} \phi} \in \mathrm{el}(\varphi)\right\} \ .$$

- Fair states:

$$\mathcal{F}^K_\varphi \doteq \{s \mid T_\varphi, s \models \mathsf{E} \bigwedge_{\psi \in \mathcal{J}_\varphi} \mathsf{GF}\psi\} \ .$$

The Symbolic Monitoring Algorithm

function symbolic_monitor$(K \doteq \langle V_K, \Theta_K, \rho_K, \mathcal{J}_K \rangle, \varphi(AP), u \in (\Psi(O) \times \mathbb{B})^*)$

- $T_\varphi \doteq \langle V_\varphi, \Theta_\varphi, \rho_\varphi, \mathcal{J}_\varphi \rangle \longleftarrow$ ltl_translation(φ);
- $T_{\neg\varphi} \doteq \langle V_\varphi, \Theta_{\neg\varphi}, \rho_\varphi, \mathcal{J}_\varphi \rangle \longleftarrow$ ltl_translation$(\neg\varphi)$;
- $\mathcal{F}^K_\varphi \longleftarrow$ fair_states$(K \otimes T_\varphi)$;
- $\mathcal{F}^K_{\neg\varphi} \longleftarrow$ fair_states$(K \otimes T_{\neg\varphi})$;
- $r_\varphi \longleftarrow \Theta_K \wedge \Theta_\varphi \wedge \mathcal{F}^K_\varphi$; /* no observation */
- $r_{\neg\varphi} \longleftarrow \Theta_K \wedge \Theta_{\neg\varphi} \wedge \mathcal{F}^K_{\neg\varphi}$;
- **if** $|u| > 0$ **then** /* first observation */
  - $r_\varphi \longleftarrow r_\varphi \wedge \text{OBS}(u_0)$;
  - $r_{\neg\varphi} \longleftarrow r_{\neg\varphi} \wedge \text{OBS}(u_0)$;
- **for** $1 \leqslant i < |u|$ **do** /* more observations */
  - **if** $\text{RES}(u_i) = \bot$ **then** /* no reset */
    - $r_\varphi \longleftarrow \text{fwd}(r_\varphi, \rho_K \wedge \rho_\varphi)(V_K \cup V_\varphi) \wedge \text{OBS}(u_i) \wedge \mathcal{F}^K_\varphi$;
    - $r_{\neg\varphi} \longleftarrow \text{fwd}(r_{\neg\varphi}, \rho_K \wedge \rho_\varphi)(V_K \cup V_\varphi) \wedge \text{OBS}(u_i) \wedge \mathcal{F}^K_{\neg\varphi}$;
  - **else** /* with reset */
    - $r \longleftarrow r_\varphi \vee r_{\neg\varphi}$;
    - $r_\varphi \longleftarrow \text{fwd}(r, \rho_K \wedge \rho_\varphi)(V_K \cup V_\varphi) \wedge \chi(\varphi) \wedge \text{OBS}(u_i) \wedge \mathcal{F}^K_\varphi$;
    - $r_{\neg\varphi} \longleftarrow \text{fwd}(r, \rho_K \wedge \rho_\varphi)(V_K \cup V_\varphi) \wedge \chi(\neg\varphi) \wedge \text{OBS}(u_i) \wedge \mathcal{F}^K_{\neg\varphi}$;
- **if** $r_\varphi = r_{\neg\varphi} = \bot$ **then return** $\times$;
- **else if** $r_\varphi = \bot$ **then return** $\bot^{\mathrm{a}}$;
- **else if** $r_{\neg\varphi} = \bot$ **then return** $\top^{\mathrm{a}}$;
- **else return** ?;

The Symbolic Algorithm: A Sample Run

Monitoring $p\,\mathrm{U}\,q$ assuming $p \neq q$

$$\varphi \doteq p\,\mathrm{U}\,q \equiv q \vee (p \wedge \mathrm{X}(p\,\mathrm{U}\,q))),$$

$$O = \{p, q\}, \qquad V_\varphi = \{p, q, x \doteq \mathrm{X}_{p\mathrm{U}q}\},$$

$$\Theta_\varphi = q \vee (p \wedge x), \qquad \Theta_{\neg\varphi} = \neg(q \vee (p \wedge x)),$$

$$\rho_\varphi = x \leftrightarrow (q' \vee (p' \wedge x')), \qquad \mathcal{J}_\varphi = \mathcal{J}_{\neg\varphi} = \top,$$

$$K = \langle O, p \neq q, p' \neq q', \emptyset \rangle, \qquad u = \{p\}\{p\}\cdots\{q\}\{q\}\cdots.$$

Executation

1. Initially (L6–7): $r_\varphi \leftarrow \Theta_\varphi$, $r_{\neg\varphi} \leftarrow \Theta_{\neg\varphi}$;
2. Taking $u_0 = \{p\}$ (L9–10): $r_\varphi = \Theta_\varphi \wedge (p \wedge \neg q) \equiv p \wedge \neg q \wedge x$, $r_{\neg\varphi} = \Theta_{\neg\varphi} \wedge (p \wedge \neg q) \equiv p \wedge \neg q \wedge \neg x$. (output is ?)
3. On next $\{p\}$, r_φ and $r_{\neg\varphi}$ remain unchanged (L13–14), as $\rho_\varphi \wedge (p' \wedge \neg q') \equiv x \leftrightarrow x'$.
4. On next $\{q\}$, $\rho_\varphi \wedge (\neg p' \wedge q') \equiv x \leftrightarrow \top$, and $\mathrm{fwd}(r_{\neg\varphi}, \rho_\varphi)(V_\varphi) \wedge (\neg p' \wedge q')$ (L14) is unsatisfiable, i.e. $r_{\neg\varphi} = \bot$. (r_φ is still not empty, output is $\top^{\mathrm{a}}$.)
5. Taking more $\{q\}$ does not change the output, unless the assumption $p \neq q$ is broken: $r_\varphi = r_{\neg\varphi} = \bot$, output is $\times$.

Correctness Proof (Sketch)

The function `symbolic_monitor` implements the monitor function $\mathcal{M}^K_\varphi(\cdot)$.

Proof.

1. Some abbreviations:

$$
\begin{aligned}
u \lesssim w \quad &\Leftrightarrow \quad \forall i.\ i < |u| \Rightarrow w_i(V_k \cup AP) \models \textsc{obs}(u_i)(O), && (1)\\
\mathcal{L}^K_\varphi(u) \quad &\doteq \quad \{w \in \mathcal{L}(K) \mid (w, \textsc{mrr}(u) \models \varphi) \land u \lesssim w\}, && (2)\\
L^K_\varphi(u) \quad &\doteq \quad \{v \mid \exists w.\ v \cdot w \in \mathcal{L}^K_\varphi(u) \land |v| = |u|\}\ . && (3)
\end{aligned}
$$

2. Reduced goal: $L^K_\varphi(u) = \emptyset \Rightarrow r_\varphi(u) = \emptyset$ and $L^K_{\neg\varphi}(u) = \emptyset \Rightarrow r_{\neg\varphi}(u) = \emptyset$.
3. Loop invariants (by induction on the length of u):

$$
\begin{aligned}
r_\varphi(u) &= \left\{s \mid \exists w \in \mathcal{L}(K \otimes T^\varphi).\ (w, \textsc{mrr}(u) \models \varphi) \land u \lesssim w \land w_{|u|} = s\right\},\\
r_{\neg\varphi}(u) &= \left\{s \mid \exists w \in \mathcal{L}(K \otimes T^{\neg\varphi}).\ (w, \textsc{mrr}(u) \models \neg\varphi) \land u \lesssim w \land w_{|u|} = s\right\}\ .
\end{aligned} \tag{4}
$$

4. Correctness of reset (Line 13):

$$
r_\varphi(u) \lor r_{\neg\varphi}(u) = \left\{s \mid \exists w \in \mathcal{L}(K \otimes T^\varphi_0).\ u \lesssim w \ \land\ w_{|u|} = s\right\} \quad \texttt{where} \quad T^\varphi_0 = \langle V_\varphi, \bigwedge_{\mathrm{Y}_p \in\, \mathrm{el}(\varphi)} \neg \mathrm{Y}_p, \rho_\varphi, \mathcal{J}_\varphi\rangle\ . \tag{5}
$$

Modifying/Extending the Algorithm

From Offline to Online Monitors

1. Preparing initial belief states r_φ and $r_{\neg\varphi}$;
2. LOOP start: taking just one input state s;
3. Update belief states, getting new r_φ and $r_{\neg\varphi}$;
4. Output a verdict in $\mathbb{B}_4$.

From Symbolic to Explicit-State Monitors

1. Utilizing the *canonicity* of BDDs: each Boolean function (up to $=$) has an unique address in the memory;
2. Constructing monitor automata for all possible inputs;
3. Each location of the automaton represent a pair of BDDs: $(r_\varphi, r_{\neg\varphi})$;
4. Always terminates due to the finiteness of BDDs.

Outline

Experimental evaluation - The tool

The monitoring algorithm has been implemented in NuRV - an nuXmv extension for Runtime Verification[3].

NuRV's features

- "Traditional" RV plus ABRV
- Offline vs. Online
- BDD-based v.s Code Generation
- Reactive vs. Deductive
- Code generation in various programming language;
- Code generation in SMV, allowing formal verifications of the correctness or other properties of the monitor.

[3] A. Cimatti, C. Tian, and S. Tonetta. NuRV: a nuXmv Extension for Runtime Verification. In B. Finkbeiner and L. Mariani, editors, *LNCS 11757 - Runtime Verification (RV 2019)*. Springer International Publishing, Porto, Portugal, Oct. 2019

Model checking SMV monitors

Used NUXMV to encode the following properties:

- The *rough* correctness (w/o resets):
  $(\mathsf{F}\ \texttt{M.\_true}) \to \varphi$ or $(\mathsf{F}\ \texttt{M.\_false}) \to \neg\varphi$;
- The monotonicity of monitors:
  $\mathsf{G}\ \texttt{M.\_unknown} \vee (\texttt{M.\_unknown}\ \mathsf{U}\ \texttt{M.\_concl})$;
- Comparison of two monitors (M1: with BI, M2: w/o BI):
  $\neg\mathsf{F}\ \texttt{BIv}$, where $\texttt{BIv} := (\texttt{M1.\_concl} \wedge \neg\,\texttt{M2.\_concl})$.
- The correctness of resets:
  $\mathsf{X}^n\,(\texttt{M.\_reset} \wedge \mathsf{X}\ (\neg\,\texttt{M.\_reset}\ \mathsf{U}\ \texttt{M.\_true})) \to \mathsf{X}^n\varphi$.

Observation

The full correctness of LTL monitors cannot be model checked by LTL itself. (Epistemic operator needed)

Tests on LTL patterns

We tested on Dwyer's 55 LTL patterns[4]:

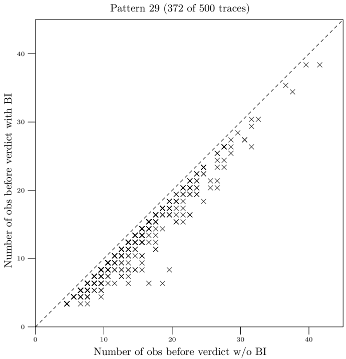
- Collected from over 500 specifications from at least 35 different sources.
- 11 groups: Absence, Existence, Bounded Existence, Universality, Precedence, Response (2-causes-1, 1-cause-2), Precedence Chain (2-stimulus-1, 1-stimulus-2), Response Chain, Constrained Chain;
- 5 scopes: Globally, Before, After, Between, After-Until.

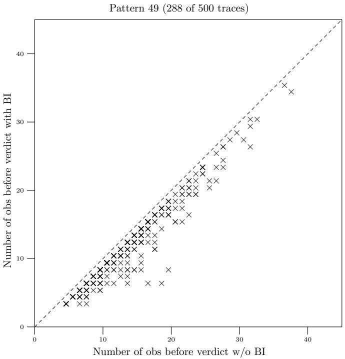
All these patterns can be successfully synthesized into working monitors (with or w/o BI); 500 random traces (each with 50 states) were used to compare the monitoring results.

[4] `https://matthewbdwyer.github.io/psp/patterns/ltl.html`

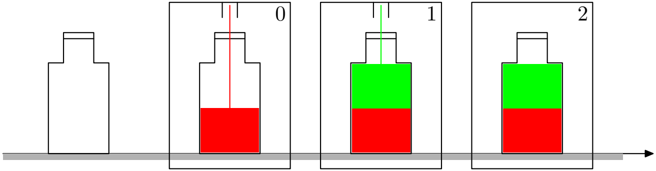
Tests - The Value of Assumption

- Assumption: transitions to s-state occur at most 2 times: $((\neg s)\,\mathsf{W}\,(s\,\mathsf{W}\,((\neg s)\,\mathsf{W}\,(s\,\mathsf{W}\,(\mathsf{G}\,\neg s)))))$
  $(\varphi\,\mathsf{W}\,\psi \doteq (\mathsf{G}\,\varphi) \vee (\varphi\,\mathsf{U}\,\psi))$
- Pattern 29: s responds to p after q until r: $\mathsf{G}\,(q \wedge \neg r \rightarrow ((p \rightarrow (\neg r\;\mathsf{U}\;(s \wedge \neg r)))\,\mathsf{W}\,r))$.
- Pattern 49: s,t responds to p after q until r:
  $\mathsf{G}\,(q \rightarrow (p \rightarrow (\neg r\,\mathsf{U}\,(s \wedge \neg r \wedge \mathsf{X}\,(\neg r\,\mathsf{U}\,t))))\,\mathsf{U}\,(r \vee \mathsf{G}\,(p \rightarrow (s \wedge \mathsf{X}\,\mathsf{F}\,t))))$





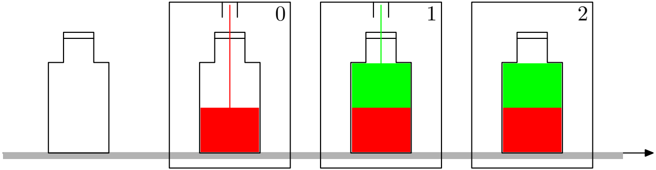
Tests on a Factory Model (1)



Model variables:

- `bottle_present[0-2]`: there exists a bottle at position 0–2;
- `bottle_ingr1[0-2]`: red ingredient in the bottle at position 0–2;
- `bottle_ingr2[0-2]`: green ingredient in the bottle at position 0–2;
- `move_belt`: the belt is moving;
- `new_bottle`: new bottle at position 0 before the belt starts to move.

Tests on a Factory Model (2)



- Whenever the belt is not moving and there is a bottle at position 2, both ingredients are filled in that bottle:

$$\varphi = \mathsf{G}\,((\texttt{bottle\_present[2]} \wedge \neg\,\texttt{move\_belt}) \rightarrow (\texttt{bottle\_ingr1[2]} \wedge \texttt{bottle\_ingr2[2]}))$$

- Two monitors: `M1` (with BI), `M2` (w/o BI).
- Model checking spec: $\neg\mathsf{F}\ \texttt{BIv}$, where $\texttt{BIv} := (\texttt{M1.\_concl} \wedge \neg\,\texttt{M2.\_concl})$.
- *Conclusion*: the monitor `M1` is predictive, it outputs $\perp^{a}$ soon after the fault at position 0.

Conclusions

- We propose ABRV – an extended RV framework where assumptions, partial observability and resets are supported;
- A new four-valued LTL semantics called ABRV-LTL extended from LTL_3;
- A symbolic PLTL monitoring algorithm for ABRV;
- Under certain assumptions: 1) the resulting monitors are predictive; 2) some non-monitorable properties become monitorable.

Future directions

- ABRV with more expressive (temporal) logics: LTL + metric/epistemic operator(s); FOL (with quantifiers over time); MSOL (WS1S, WS2S);
- MC/SAT-based monitoring algorithms (when BDD is not applicable).

Question (independent of PLTL)

How to support Assumption/Partial-Observability/Resets in your RV tool/approach?

Backup: Fair Kripke Structure (FKS)

Let $\mathbb{B} = \{\top, \bot\}$ denote the type of Boolean values, a set of *Boolean formulae* $\Psi(V)$ over a set of propositional variables $V = \{v_1, \ldots, v_n\}$, is the set of all *well-formed formulae* (wff) built from variables in V, propositional logical operators like $\neg$ and $\wedge$, and parenthesis.

Definition

Let V be a set of Boolean variables, and $V' \doteq \{v' \mid v \in V\}$ be the set of *next state* variables (thus $V \cap V' = \emptyset$). An FKS $K = \langle V, \Theta, \rho, \mathcal{J} \rangle$ is given by

- V, the set of Booleam variables,
- a set of initial states $\Theta(V) \in \Psi(V)$;
- a transition relation $\rho(V, V') \in \Psi(V \cup V')$,
- a set of Boolean formulae $\mathcal{J} = \{J_1(V), \ldots, J_k(V)\} \subseteq \Psi(V)$ called *justice requirements*.

The *forward image* of a set of states $\psi(V)$ on $\rho(V, V')$ is a Boolean formula $\mathrm{fwd}(\psi, \rho)(V) \doteq (\exists V.\, \rho(V, V') \wedge \psi(V))[V/V']$, where $[V/V']$ substitutes all (free) variables from V' to V.